# Operating System Tutorial Exam Solutions

1) Which signal is generated when we press ctrl-Z?
   a) SIGKILL b) ***SIGSTOP / SIGTSTP*** c) SIGABRT d) SIGINT

2) The ***Mode*** argument of ***Creat()*** system call is defined in the_____header file
   (a**)<sys/stat.h>** (b) <fcntl.h> (c) <sys/types.h> (d) <stdio.h>

3) The meaning of the ***lseek(fd, 0, 2)*** system call is
   (a) ***places the current file pointer at the end-of-file*** (b) places the current pointer after the second byte of the file (c) places the current pointer at the first byte (d) backs up the file pointer by two bytes from the end-of-file

4) Default activity of SIGSEGV is
   a) Terminate b) **Core dump** c) Stop d) Cont

5) Kernel modules are available in
   a**) /lib directory** b) /root directory c) /boot directory d) none of the mentioned

6) At the point when a process is forked the child process inherits a few properties. Which of the following properties are inherited?
   a) Timers **b) CWD, Root Directory** c) PID d) **Shared Memory Segments**
   e) Asynchronous I/O Operations f) **Resource Limits g) open file descriptors**

7) What number of process will be spawned in the wake of executing the below system?

```
#include <stdio.h>
#include <unistd.h>
int main() {
fork();
fork() && fork() || fork();
fork() && fork();
printf("BITS\n");
return 0;  }
```

***Sol: 29 or 30***

8) Expect the output of the below code?

```
void my_signal_handler ( int signalnum) {
   printf("Its Done\n");    }
int main() {
   int pid;
```

```
signal (SIGKILL, my_signal_handler);
pid = fork();
if (pid!=0) {
    sleep(10);
} else {
   kill(getppid(), SIGKILL);
   exit(0);
}
return 0;    }
```

**Sol:** ***No Output*** **or** ***Parent exits without going to the signal handler***

9) Modify the below code such that it prints a message "***I am ignoring the signal*** " immediately when you provide an first interrupt (Ctrl+c) and the process should be terminated on providing the next (Second) interrupt (Ctrl+c). Write the corresponding function and use proper signals for the same.

```
#include<stdio.h>
#include<signal.h>

int main()  {
      while(1){
       printf("BITS\n");
       sleep(5);
     }
     return 0;      }
```

Sol:

```
1.    #include<stdio.h>
2.    #include<signal.h>
3.
4.    void signal_func (int);
5.    void signal_func (int sig_no)
6.    {
7.       printf("I am ignoring the signal \n");
8.       signal(SIGINT,SIG_DFL);
9.    }
10.   int main()
11.   {
12.      signal(SIGINT, signal_func);
13.      while(1){
14.         printf("BITS \n");
15.         sleep(5);
16.      }
17.      return 0;
```

```
int count =0;
void signal_func (int sig_no)
  {
     if (count ==0)
     {
     printf("I am ignoring the signal \n");
     count++;
     }
    else
      exit(1);
  }
```

18. }

10) Predict the output of the below code snippet ?

```
#include <stdio.h>
#include <fcntl.h>
int main() {
      int fp, counter;
      char charac[10];
      fp = open("bitspilani.txt",O_RDWR|O_CREAT);
      write(fp,"operating systems",5);
      lseek(fp,2,SEEK_END);
      write(fp,"tutorial",3);
      lseek(fp,0,0);
      counter = read(fp,charac,10);
      printf("%s\n",charac);
      return 0;
  }
```

Sol: ***opera***

11) Make necessary changes in the code which pipes the output of "ls" command to "grep" command for listing out ".py" files in the current directory.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define COMND1 "grep"
#define COMND2 "ls"
int main() {
    int Pipe[2];
    int myStdin, myStdout;
    int pidchild;
    myStdin = dup(0);
    myStdout = dup(1);
    pipe(Pipe);
```

```
    if( ! pidchild ) {
       dup2( myStdout, 1 );
        close( myStdout );
        close( myStdin );
        close( Pipe[0] );
        close( Pipe[1] );
        printf("\n");   }

    close( Pipe[0] );
    close( Pipe[1] );
    close( myStdout );
    close( myStdin );
     return 0;
}
```

Sol:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define COMND1 "grep"
#define COMND2 "ls"
```

***#define ARG1 ".py"***

```
int main()
{
   int Pipe[2];
   int myStdin, myStdout;
   int pidchild;

   myStdin = dup(0);
   myStdout = dup(1);

   pipe(Pipe);
   if( (pidchild = fork()) < 0 )
   {
      perror( "Fork Error" );
      exit(1);
   }

   if( ! pidchild )
   {
       dup2( Pipe[0], 0 );

      dup2( myStdout, 1 );

      close( myStdout );
      close( myStdin );
      close( Pipe[0] );
      close( Pipe[1] );

      printf("\n");
          execlp( COMND1, COMND1, ARG1, NULL );
   }

   dup2( Pipe[1], 1 );

   close( Pipe[0] );
   close( Pipe[1] );
   close( myStdout );
   close( myStdin );
   execlp( COMND2, COMND2, NULL );
   return 0;
}
```